Expert System Technology in Observing Tools

Karl Wolf [c], Chris Burkhardt [a], Mark Fishman [c], Sandy Grosvenor [d], Jeremy Jones [b], Anuradha Koratkar[a], and LaMont Ruley [b]

[a]Space Telescope Science Institute, 3700 San Martin Drive, Baltimore, MD 21218
[b]NASA Goddard Space Flight Center, Greenbelt, MD 20771
[c]AppNet Inc., 1100 West Street, Laurel, MD 20707
[d]Booz-Allen Hamilton, 7404 Executive Place, Seabrook, MD 20706

ABSTRACT

Over the past two years, the Scientist's Expert Assistant team from NASA's Goddard Space Flight Center and the Space Telescope Science Institute has been prototyping tools to support General Observer proposal development for the Hubble Space Telescope and the Next Generation Space Telescope. One aspect of this effort has been the exploration of the use of expert systems in guiding the user in preparing their observing program. The initial goal was to provide the user with a question-and-answer style of interaction where the software would "interview" the user for their science needs and recommend instrument settings. This design ultimately failed. The reasons for this failure, and the resulting evolution of our approach, are an interesting case study in the use of expert system technology for observing tools. Although the interview approach failed we felt that expert systems can still be used in the tools environment. This paper describes our current approach to the use of expert systems and how it has evolved over the project's lifetime. We also present suggestions on why expert systems are useful and when they are appropriate.

Keywords: expert system, SEA, Scientist's Expert Assistant, rulebase, rule engine, proposal preparation

1. INTRODUCTION

The Scientist's Expert Assistant (SEA)[1] is a prototype tool designed to support the observing proposal development for the Hubble Space Telescope. The purpose of the SEA is to investigate automated solutions for reducing the time and effort involved for both scientists and telescope operations staff spent in preparing detailed observing proposals.

The SEA is being developed by a collaborative effort between the Advanced Architectures and Automation Branch of the NASA Goddard Space Flight Center and the Space Telescope Science Institute. The tool has been under development for about two years and expert system technology has been a part of the effort from the beginning.

Early in the development of the SEA, we recognized there was a great deal of knowledge, both astronomical and procedural, required by an observer to create an observation proposal. From the beginning, the SEA's primary goal was to make it easier to create both Phase I and II observing proposals. We therefore felt it would make the development process easier if we could build this expert knowledge into the SEA. The intent was to aid the novice user while acting as a double-check for the experienced observer. However, it has not been easy to determine the best way to integrate the expert system technology into the SEA. We have made three attempts with varying degrees of success. From our evaluation (see A. Koratkar[2]) we have found that it is important that further research continue with expert systems because the users constantly indicated the difficulty of understanding the complexity of the new instruments. The users need to have the confidence to feel comfortable with their final program even when they know intutively that their scientific program is very simple.

2. A BRIEF OVERVIEW OF EXPERT SYSTEM TECHNOLOGY

Expert System Technology is an applied branch of Artificial Intelligence research, which focuses on techniques to incorporate detailed domain expert knowledge into software systems. The knowledge is usually embodied in something called a "rulebase", which is a collection of rules, and the data used by those rules. The rules model the domain logic (aka "Business Logic") and usually have a simple IF-THEN-ELSE format. They are intended to be readable (they say) by non-programmers. The data is typically used to maintain context and state information.

[1] The SEA web site can be found at: http://aaaprod.gsfc.nasa.gov/sea

A rule conditionally maps one or more pieces of data to a set of actions to perform if the conditions are met (or not met). The IF portion of a rule defines the condition. The “THEN” portion defines the set of actions to take when the condition evaluates to TRUE, and the optional “ELSE” portion defines the set of actions to otherwise take. The rule defines how data will change based on the current state of the data. For example:

```
Rule temp_warning is:
if  current_temperature > Max_Temp then
{
      color = “red”.
}
else
{
      color = “green”.
}
```

The name of the rule is **temp_warning**. In the condition clause, the data item **current_temperature** is compared to another data item called **Max_temp.** When the comparison evaluates to TRUE then the **color** data item is set to the value, “red”. Since **color** is a data item, other rules could depend on it. Those rules may subsequently be processed.

All rule processing is handled by a software package called a “rule engine”. As data changes and external events occur, the rule engine determines which rules are eligible for firing because their condition clause becomes true. These rules are placed on a list called the Agenda. Once all of the eligible rules have been determined, the rules on the agenda are processed, or “fired”, one-by-one. Note: once a rule fires, other rules may be added or removed on the agenda as a result of changes incurred by the just fired rule. The processing continues until all of the rules on the agenda have been processed. The order of processing is generally not defined. This general rule and agenda processing is known as the RETE-Algorithm, and it is probably the most common algorithm used by rule engines today. The algorithm is fast and efficient.

Expert Systems generally excel at handling situations where the current state of the data is incomplete or unknown, which is perfect for a multi-parameter proposal development effort. A variable can be compared to a special case value such as “Unknown”. Rules can be defined that depend on missing data. They typically prompt the user to ask for more data.

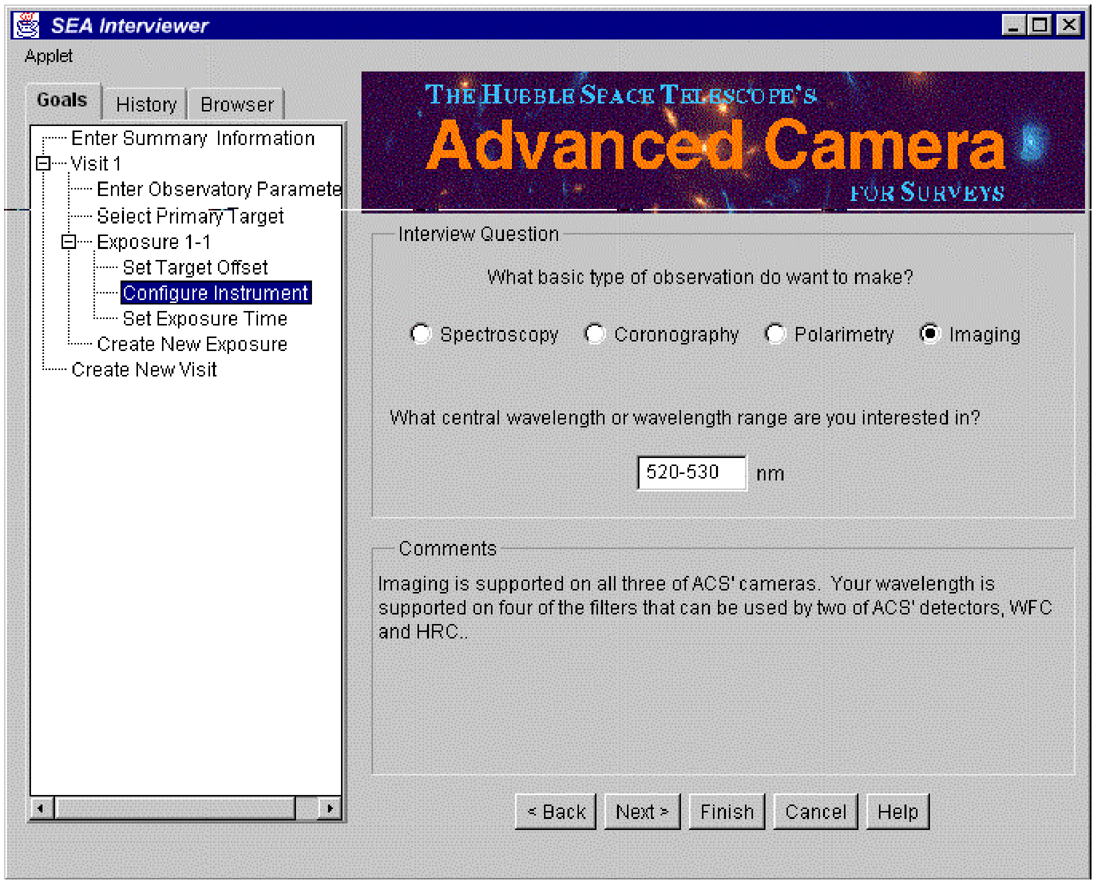
The expressed purpose of Expert System Technology seemed to match well with the purpose for the SEA. We hoped to capture at least some astronomical domain knowledge of various expert users and build into the SEA. The SEA could use that knowledge to help the user in a variety of ways.

3. FIRST ATTEMPT-INTERVIEW MODE

The original idea was to use an Expert System as the underlying foundation of the SEA. It would work behind the scenes to monitor the proposal development process. The general paradigm was modeled after the tax form interview mode in a familiar program such as MacInTax or TurboTax . In those programs, the user interacts with an application that is an “expert” in the income tax preparation domain. As we are sure you are familiar, income tax forms are complex with a great amount of data and many intertwining rules and special cases.

The income tax programs have two primary modes of operation. First, they provide direct access to the forms, letting the user directly control what to do next. The programs handle the propagation of values and calculations as needed, much like a spreadsheet. The second is an interactive Interview mode. The programs guide the user through the potentially lengthy tax preparation process through question and answer sessions. The determination of the next step is based on the user’s answers to previous steps combined with the domain knowledge of tax preparation process. The programs ensure the data provided by the user is consistent and “makes sense”.

We initially viewed the proposal development process to be similar to the tax form preparation process. The SEA expert system rulebase would present one or more questions to the user, monitor the user’s responses, then generate appropriate (insightful?) questions and provide feedback in the form of comments. The expert system would act as a guide through the process. Figure 1 below shows an example of an interview page.



Interview Question

What basic type of observation do want to make?

Spectroscopy Coronography Polarimetry [✓ Imaging]

What central wavelength or wavelength range are you interested in?

520-530 nm

Comments

Imaging is supported on all three of ACS' cameras. Your wavelength is supported on four of the filters that can be used by two of ACS' detectors, WFC and HRC..

Figure 1: Interview Mode

Our prototype of this paradigm was not very successful due to a variety of reasons:

- The somewhat rigid interview sequencing tended to hinder exploration. The user had to follow prescribed paths through the interview process.

- The questions often didn't fit well with what the user was thinking. The users were generally focused on the science, not on forms entry or the structure of a proposal. It was often difficult to anticipate what the user might want to do next.

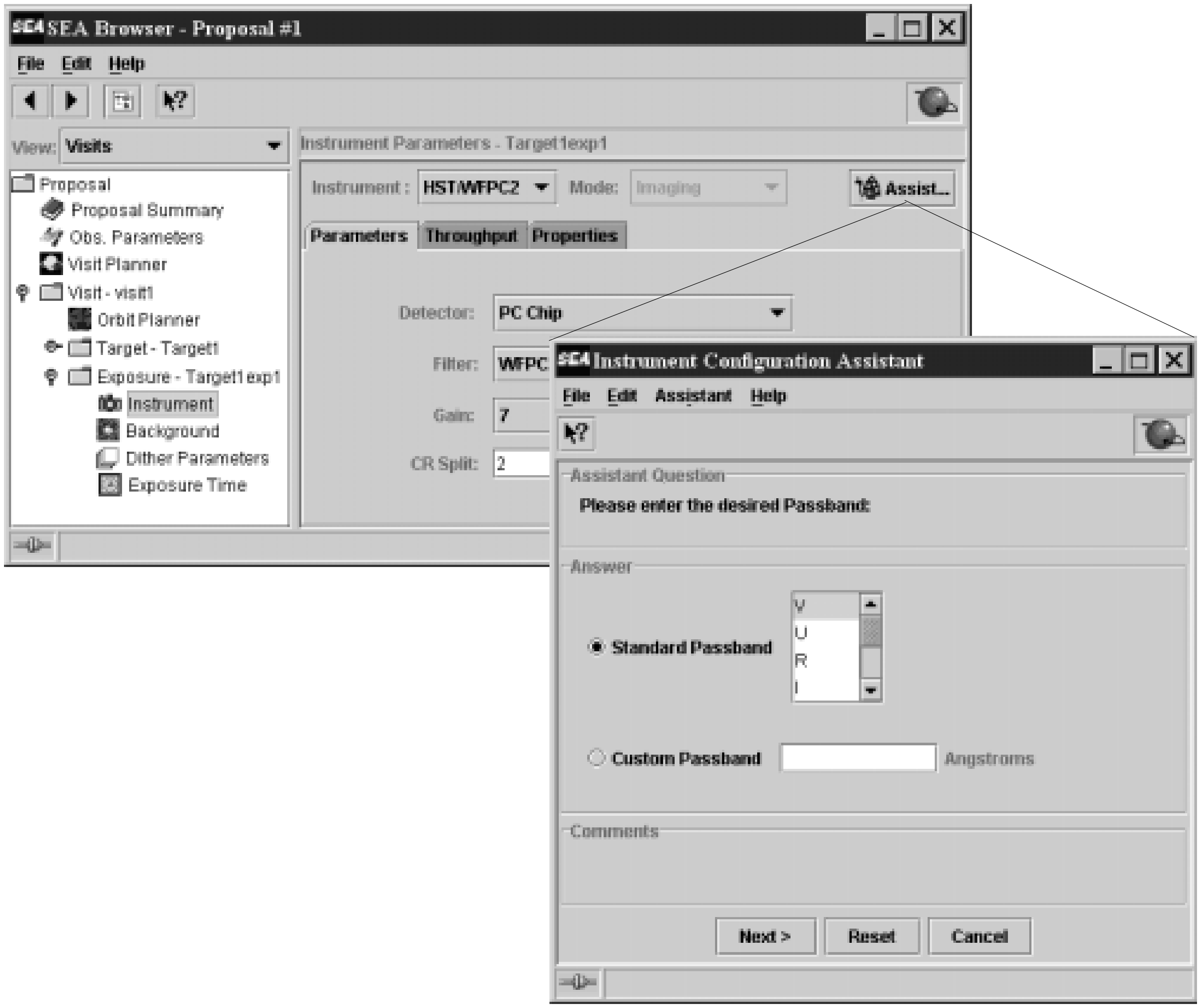
- An interview paradigm was not appropriate for every portion of the proposal development process. For example; the user might want to set an exposure time using the Exposure Time Calculator (ETC), not answering the next interview question.

- The underlying expert system engine technology was immature, resulting in a single monolithic and overly complex rulebase. The rules became intertwined with lots of flags and conditionals to control the flow of rule execution and to maintain processing state. As SEA grew, the rulebase became unwieldy and slow. The rulebase would only get more unwieldy, slower, and more complex as new features and capabilities were added to the SEA.

The all-inclusive Interview Mode was abandoned during the early stages of the SEA prototype development.

4. SECOND ATTEMPT – THE ASSISTANT

We dramatically scaled back the scope of the expert system technology in our second integration effort. Rather than use the expert system as a primary foundation technology underlying all of the SEA, we relegated the expert system to a small portion of the application. Specifically, we used the expert system as an “assistant” to help the user with exploration, for example which is the best instrument/detector/filter combination for my scientific needs? The expert system only became active when the user selected the "Assist" button on the Instrument Configuration panel. In other words, the user had to request the assistance from a dormant expert system. Figure 2 below shows the first assistant panel.



Instrument : HST/WFPC2
Mode: Imaging
Detector: PC Chip
Filter: WFPC
Gain: 7
CR Split: 2

Please enter the desired Passband:

(●) Standard Passband: V, U, R, I
( ) Custom Passband: Angstroms

Figure 2: Instrument Configuration Assistant

The expert system was idle most of the time and only became active when requested by the user. The Expert System would wake up and prompt with two questions: (what passband? as seen above and what type of observation?). The Expert System would traverse through its list of combinations in its knowledge base and assign a "goodness" factor to each of them. The “goodness” factor was a numerical value computed from the filter’s effective wavelengths and how well they compared to the specified passband. The combinations were sorted by their goodness factors and presented to the user for selection. This strategy was highly appreciated by many of the evaluators as they saw its use during “Phase 0” and for multi-observatory exploration purposes. For many other users who focused their evaluation of SEA as a Phase II tool, this strategy was considered okay but not useful.

The rulebase became very small and algorithmic due to its limited responsibilities and the mechanical way we chose to process the filters. During the rulebase development, the vendor supplied rule engine became more powerful and better able to control rule processing flow. We no longer needed a lot of data to maintain the state and to control the rule flow execution as we saw in our first integration attempt (see section 3 First Attempt-Interview Mode above). The reduced complexity caused a complementary reduction in rulebase size.

The SEA is implemented in the Java programming language. One of the built in features of Java is support for application Threads, one or more parallel execution paths within a single application.
 below diagrams how the Instrument Configuration Assistant Thread was implemented within the SEA.

The sequencing is as follows:

- The user requests assistance by selecting the **Assist** button on the Instrument Configuration panel.
- The rulebase resolves Unknown data by asking the user for additional input (see Figure 2).
- The rule engine applies the rulebase rules.
- Results are loaded into a Results array.
- The user views the Results array and usually selects one of the choices.
- The Main SEA Thread continues.

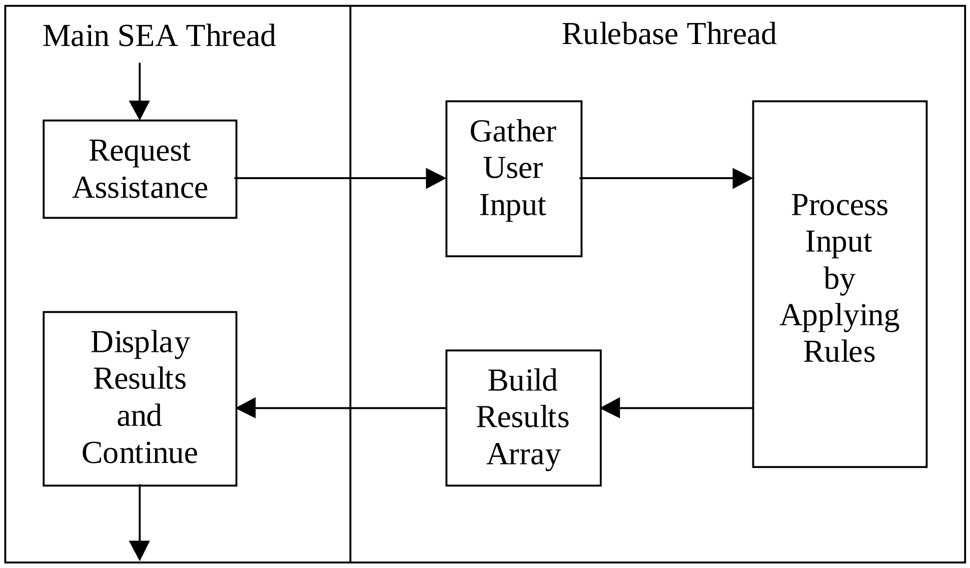


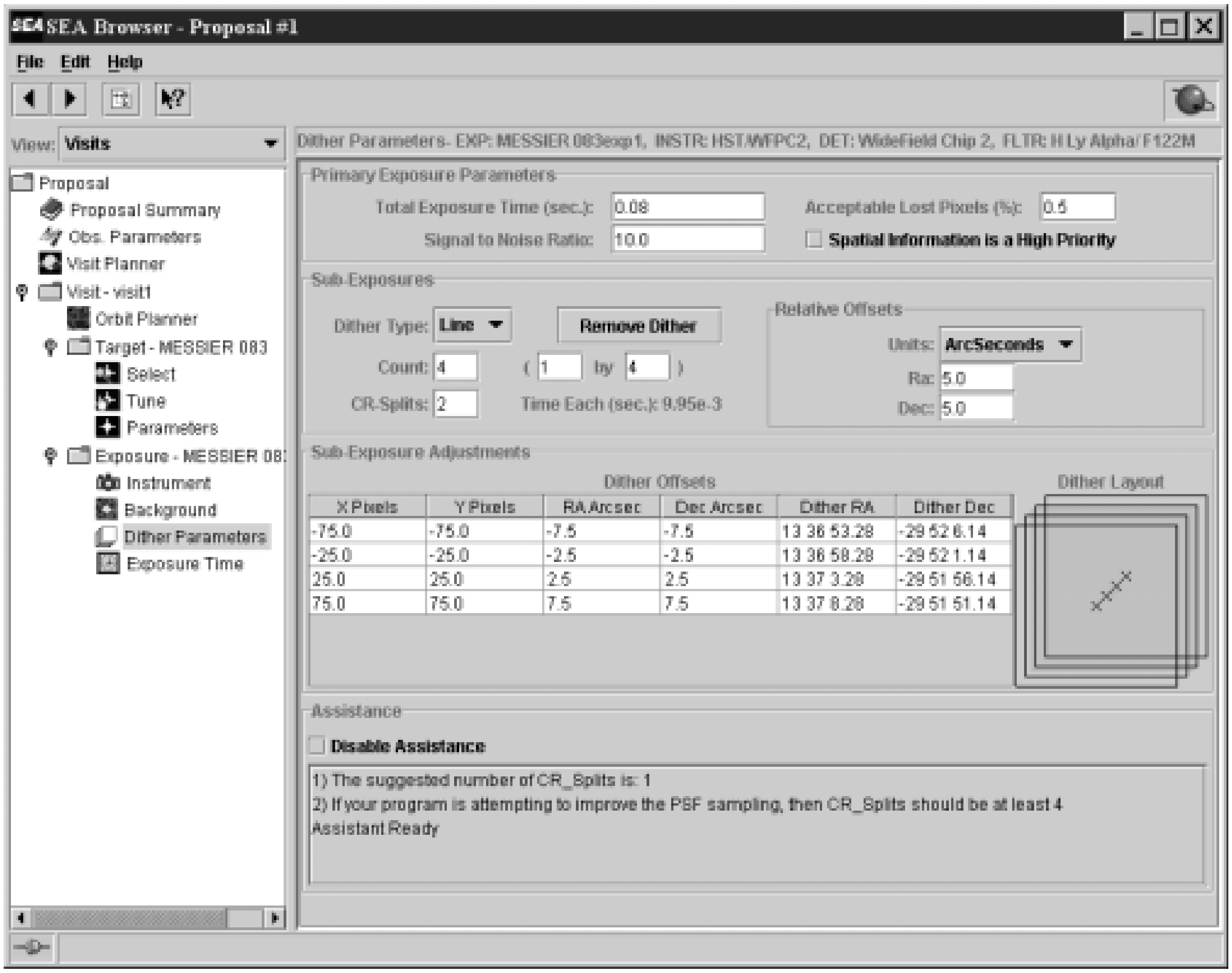
Figure 3: Instrument Configuration Assistant Rule Processing Flow

This processing is not a very effective use of Threads. The above processing could just as well have been written in straight Java code and handled as a sequence of method calls, thereby reducing the runtime overhead for rule processing. Rulebases are loaded and interpreted at runtime by the Rule Engine. The Rule Engine we used is a large and complex collection of Java classes and it often takes many seconds to load the classes and run the rule engine. Subsequent executions are much quicker because they do not incur the class loading and rule compiling costs, but that initial load can be frustrating for the user.

While the Assistant is still a part of the SEA, we felt we could do more, which brings us to our third attempt at integrating expert systems into the SEA.

5. THIRD ATTEMPT – HELPFUL OBSERVER

The third attempt is somewhere between the two previous attempts. One of the last features we added to the SEA was a panel for specifying an Exposure’s Dither Pattern (see Figure 4 below). In this module, the rulebase passively watches the user's changes and offers suggestions and warnings as appropriate. The user is free to explore while the Expert System analyzes state changes and makes suggestions in a small text area at the bottom of the module. This attempt is similar to our first attempt, but not as pervasive and it doesn't force itself on the user. We understand that our users are typically human with a variety of human temperaments, therefore we included a mechanism for the user to completely disable all comments and suggestions. We felt is was important to avoid annoying users with too much unrequested help. Although this was the least developed of the SEA's functionalities, the evaluators thought that such a functionality would be useful once it is fully developed.



Dither Parameters- EXP: MESSIER 083exp1, INSTR: HST/WFPC2, DET: WideField Chip 2, FLTR: H Ly Alpha/ F122M

Primary Exposure Parameters

Total Exposure Time (sec.): 0.08
Acceptable Lost Pixels (%): 0.5
Signal to Noise Ratio: 10.0
Spatial Information is a High Priority

Sub-Exposures

Dither Type: Line
Count: 4
(1 by 4)
CR-Splits: 2
Time Each (sec.): 9.95e-3

Relative Offsets

Units: ArcSeconds
Ra: 5.0
Dec: 5.0

Sub-Exposure Adjustments

Dither Offsets					
X Pixels	Y Pixels	RA Arcsec	Dec Arcsec	Dither RA	Dither Dec
-75.0	-75.0	-7.5	-7.5	13 36 53.28	-29 52 6.14
-25.0	-25.0	-2.5	-2.5	13 36 58.28	-29 52 1.14
25.0	25.0	2.5	2.5	13 37 3.28	-29 51 56.14
75.0	75.0	7.5	7.5	13 37 8.28	-29 51 51.14

Assistance

Disable Assistance

1) The suggested number of CR_Splits is: 1
2) If your program is attempting to improve the PSF sampling, then CR_Splits should be at least 4
Assistant Ready

Figure 4: Dither Pattern Specification Panel

Like the SEA, the underlying Expert System Technology also advanced since our previous attempts. Two key improvements enabled us to create a more effective implementation.

1. Vastly improved asynchronous event handling. The Java rule engine we used supported rules firing when an instance of an object was created, deleted, initialized, or when specific properties of an object changed or was required. In addition, the main Java SEA application could be notified when the rule engine created new objects. (See Figure 5 below) The combination of all these asynchronous event mechanisms enabled the rule engine to monitor events occurring in the main SEA thread and to report rule processing results as they happened.

2. The rulebase became much more modularized and efficient. In previous versions, all rulebases had to be a single monolithic rule file. All rules were stored together and evaluated together. This was very inefficient and

hard to program. The improvement was the implementation of something called a "ruleset". A ruleset is a grouping of rules into logical collections. The rulesets enable the programmer to control which groups of rules to apply for a given context. Only those rules necessary in the current context need to be evaluated. By segregating rules that do not apply, you avoid possible unintended interaction among all the rules and you increase efficiency.

These capabilities helped to eliminate the monolithic rulebase problem as seen during our first attempt. In addition, since we don't force a dialog with the user we avoided the exploration problems also seen in our first attempt.

When the Dither module first starts up, it creates a separate rulebase thread. This new thread loads the rule engine Java class code and the Dither rulebase. While that thread is initializing, the main Dither module continues in the main thread to build and display itself. The Dither module then accepts user input. The rule processing flow is as follows: (See Figure 5 below)

1. Every time the user changes a value in the Dither module, a new DitherContext object is created and passed to the rulebase thread.
2. The Dither module continues to accept other input.
3. In the meantime, the rulebase thread extracts the necessary information from the DitherContext and applies the Dither rules to that data.
4. During processing, one or more DitherResult objects may be created by the rulebase.
5. Every time a DitherResult object is created, it is asynchronously transferred back to the main thread. The main thread extracts and displays the result information.
6. Once all rule processing has completed, the rulebase goes back to step 1.

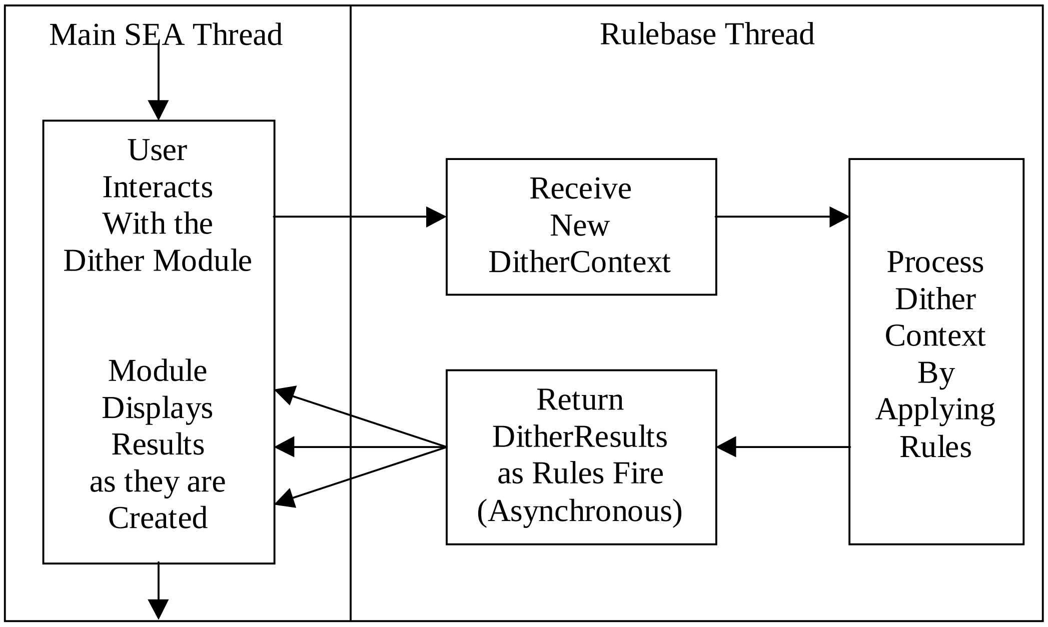

Main SEA Thread
Rulebase Thread
User Interacts With the Dither Module
Module Displays Results as they are Created
Receive New DitherContext
Process Dither Context By Applying Rules
Return DitherResults as Rules Fire (Asynchronous)

Figure 5: Dither Pattern Rule Processing Flow

The difference between this processing flow and the one in Figure 3 is that the main thread continues to operate while the rule engine is working. At any time, the rule engine can communicate any desired information back to the main thread simply be creating a new DitherResults object. The Dither rulebase is essentially a controlled passive observer of the main Dither module. It can provide guidance only when permitted to do so, and only in a quiet and unobtrusive way.

6. FUTURE DIRECTIONS

Currently an expert system is used in only two small areas in the SEA, in Instrument Configuration (Second Attempt – The Assistant) and in Dithering (Third Attempt – Helpful Observer). We would like to explore a variety of research efforts to determine other useful and appropriate ways to utilize expert systems. We would like to assist the user (if desired) in making decisions based on some basic user provided information. To accomplish this, expert system technology will be used to evaluate the observing model and the options available to the observer. The SEA will provide recommendations not only for the observing strategy, but also for optimizing the observation as well. As a first step, the rulebase will contain rules on how to avoid common observing problems. It can alert the user and suggest best practices to avoid those problems.

We see two primary areas for the expanded use of expert systems in the SEA. We list some possible applications under each area.

1. Enhanced Assistance

    - Smart Help – an improved context sensitive help system that can do more than simply provide help for a specific field or input box. A "Smart Help" feature would have an underlying rulebase that is evaluating the overall context of the developing proposal and provide more information that is specific and relevant to the context. When the user asks for help, the expert system could determine what help might be most appropriate and make it available. More like discussing an issue with a collaborator or observatory expert

    - Add "Assist" buttons to more of the SEA modules. The assist button is used to request module specific expert system guidance through some portion of the application.

    - Intelligent and unobtrusive monitoring of the proposal so that helpful suggestions can be obtained on demand. The suggestions and the type of help should adapt to the expertise of the user. This feature is particularly helpful to first time or infrequent observers

    - Improved knowledge of each observatory/instrument/detector, their capabilities, the kinds of observations they are "best" at. Make suggestions based on this knowledge. This functionality could be very useful during the initial exploration of an observing program.

2. Functional

    - An expert system could provide intelligent automatic layout processing in Orbit Planner (Best fit, optimal placement)

    - On demand, the expert system should be able to analyze the current state of the whole proposal, and provide suggestions for optimizing the observing plan. We liken this to final run-through that tax preparation programs like TurboTax or MacInTax provide. The expert system could highlight rough areas or areas that might cause some problems with scheduling, and would ensure that nothing the user specifies can adversely affect the health and safety of the observatory

    - When paired with the Visual Target Tuner (VTT) portion of the SEA, an expert system could provide suggestions on optimal Orientation angles.

    - Suggestions on most efficient use of parallel exposures in the context of HST observing.

    - Determination of effective and efficient mosaic patterns for survey observations.

The expert system is there to guide the user and to provide "expert" advice when asked. The user must be able to easily bypass or ignore any and all expert system help. The TurboTax program is a good model. It provides expert help, without forcing you to use it. Tools with expert systems incorporated in them will save both the observer and observatory staff time, because observers can find help as soon as they need it, and user support can be provided with much fewer staff. The result is more cost effective use of everyone's time and effort.

7. CONCLUSIONS AND LESSONS LEARNED

Contrary to what expert system vendors say, rulebases need to be written by programmers. Rulebases use program constructs (variables, conditionals, subprograms, etc.) and are often interrelated and require a great deal of design and planning. Individual rules may be easy to define but they are often hard to integrate into a coherent rulebase. We had originally thought that the scientists could update these rules themselves, but this soon became unrealistic. We discuss this further in the paper by A. Koratkar, et al[1], found elsewhere in these proceedings.

We often had a difficult time determining whether a rule should be implemented as a rule in a rulebase or as normal Java code. Even after writing a few rulebases, the answer to this is still not obvious.

Overall, expert system technology shows great promise. It is expected that expert systems will have an expanded role within the SEA. We feel the combination of the Helpful Observer along with the Assist button technique demonstrate good uses of expert systems. The support technology now exists so that in the future we can continue to expand the role of the expert system into such areas as: improved context sensitive help, intelligent automatic optimized exposure layout processing in the Orbit Planner, and efficient and effective orientation and mosaic pattern techniques in the Visual Target Tuner.

8. ACKNOWLEDGEMENTS

SEA is a joint effort between Space Telescope Science Institute (STScI) and the Advanced Architectures and Automation Branch of Goddard Space Flight Center's (GSFC) Information Systems Center.

9. REFERENCES

1. A. Koratkar, et al, "Lessons Learned from the Scientist's Expert Assistant project", *Proceedings of SPIE Vol. 4010*, 2000.
2. A. Koratkar, et al, "NGST's Scientist's Expert Assistant: Evaluation Plans and Results", *Proceedings of SPIE Vol. 4010*, 2000.
3. J. Jones, et al, "Next Generation User Support Tools", *Journal of the Brazilian Society of Mechanical Sciences, Vol. XXI*, pp. 84-89, 1999.
4. A. Koratkar and S. Grosvenor, "The Next Generation User Support Tools", *Astronomical Society of the Pacific Conference Series, Volume 172*, pp. 57-64, 1998.
5. T. Brooks, et al, "An Expert Assistant System to Support the General Observer Program for NGST", *Proceedings of SPIE Vol. 3349*, pp. 450-455, 1998.
6. T. Brooks, et al, "Visualization Tools to Support Proposal Submission", *Proceedings of SPIE Vol. 3349*, pp. 441-449, 1998.
7. S. Grosvenor, et al, "Exploring AI Alternatives in Support of the General Observer Program for NGST", *Proceedings of the International Workshop on Planning and Scheduling for Space Exploration and Science*, 1997.